Europäisches
Patentamt

European
Patent Office

Office européen
des brevets

# Bescheinigung

# Certificate

# Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

**Patentanmeldung Nr.** **Patent application No.** **Demande de brevet n°**

00480005.8

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

Hatten

I.L.C. HATTEN-HECKMAN

DEN HAAG, DEN
THE HAGUE, 30/03/00
LA HAYE, LE

Europäisches Patentamt | European Patent Office | Office européen des brevets

# Blatt 2 der Bescheinigung
# Sheet 2 of the certificate
# Page 2 de l'attestation

Anmeldung Nr.:
Application no.: 00480005.8
Demande n°:

Anmeldetag:
Date of filing: 06/01/00
Date de dépôt:

Anmelder:
Applicant(s):
Demandeur(s):
INTERNATIONAL BUSINESS MACHINES CORPORATION
Armonk, NY 10504
UNITED STATES OF AMERICA

Bezeichnung der Erfindung:
Title of the invention:
Titre de l'invention:
Interleaved processing system based upon the three structure in a network router

In Anspruch genommene Prioriät(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)

| Staat:<br>State:<br>Pays: | Tag:<br>Date:<br>Date: | Aktenzeichen:<br>File no.<br>Numéro de dépôt: |
|---|---|---|

Internationale Patentklassifikation:
International Patent classification:
Classification internationale des brevets:
/

Am Anmeldetag benannte Vertragstaaten:
Contracting states designated at date of filing: AT/BE/CH/CY/DE/DK/ES/FI/FR/GB/GR/IE/IT/LI/LU/MC/NL/PT/SE
Etats contractants désignés lors du depôt:

Bemerkungen:
Remarks:
Remarques:

# INTERLEAVED PROCESSING SYSTEM BASED UPON THE TREE STRUCTURE IN A NETWORK ROUTER

## Technical field

The present invention relates generally to the systems for processing the routing and filtering information of each frame received in a router at a node of a data transmission network, and relates in particular to an interleaved processing system based upon the tree structure in a network router.

## Background

Today data communication systems are more based upon data transmission networks wherein routers are used to link remote sites. But routing is still considered to be one of the major bottlenecks in these systems essentially due to the processing time and the required memory. Even if, within the past years, routers are sometimes replaced by switches, routing functions are still needed, at least at boundaries.

A first main routing function is the determination of a routing path across the network using specific protocols. The path determination is based on a variety of metrics such as the delay introduced by the network or the link cost. In addition, this determination takes into account other rules generically called filtering, such as communication restrictions or priority criteria.

A second routing function is frame forwarding, that is to say the processing of inbound data and the subsequent forwarding of these data to the appropriate outbound destination.

In the case of routers, both functions, the determination of the routing path and the frame forwarding based upon the destination address field in the frame header, are performed within the same device. Nevertheless, new techniques tend to exploit the difference between these functions, separating the corresponding operations. For instance, a single routing path processing unit could support several frame forwarding units.

As seen before, the processing time is relatively high and is strongly variable from one routing computation to another one. It is therefore difficult to support many time sensitive applications such as multimedia.

One critical time consuming operation is the searching. This operation can be seen as the retrieval of routing information located in the pattern, in particular the destination of the data corresponding to this pattern. The searching involves essentially comparisons between a part of this pattern, or sequence, and predetermined bits series, or keys, which identify appropriate routing information. For this reason, efforts have been made to optimize the speed of comparison by using parallel processing but this method admits its own limitations.

Further to the routing function, the router has to perform a filtering process using the source address field in the frame header. Today, the routing process and the filtering process are done one after the other by using a single processing entity or simultaneously by using two separate processing units. For each process, there are two phases which are repeated until the end of the process, an instruction loading phase and an instruction processing phase. It must be noted that, in classical systems, the two phases of loading and processing are very close in duration.

Today, the routing function as well as the filtering function are using a longest matching prefix algorithm which is generally implemented in a tree structure. The filtering process may also use the same type of process and algorithm but in a separate tree structure. The trees are implemented in a memory structure containing one instruction by tree node and having each instruction providing the link to the sub-nodes from the tree root down to the leaf nodes which provides either the routing information or the filtering rules. This last information is very often given in an indirect way by an address value that corresponds to the field that contain the routing or filtering information for each leaf node. As an example, such a method is disclosed in the article "Routing on longest-matching prefixes", IEEE/ACM transactions on networking, vol. 4, n_1, February 1996, pages 86-97.

Now, the systems currently available for processing the routing function or the filtering function by using a tree structure require a lot of memory for storing the information in each node and a large number of memory accesses. Therefore, there is a need of a processing system enabling to optimize the processing of both routing and filtering functions.

**Summary of the invention**

Accordingly, the object of the invention is to provide a processing system enabling both routing function and filtering function to be processed fast by a single processing engine.

The invention relates therefore to an interleaved processing system in a network router wherein the frame to be routed includes a header containing a source address pattern and a destination address pattern, each pattern being processed by a task processor according to a tree structure starting from a root node and being made of a plurality of nodes linked by branches in which each node is associated with an instruction and has at least two child nodes linked to the parent node, the tree processing being performed at each node through the application of the instruction associated with each node and whose parameters depend on the node for determining which branch is to be taken in accordance with the pattern. The system comprises a first bank of registers for loading the instruction to be used by the task processor at each node of the tree corresponding to the source address pattern, a second bank of registers for loading the instruction to be used by the task processor at each node of the tree corresponding to the destination address pattern, and a task scheduler for generating successive alternate even and odd time cycles and for enabling the first bank of registers to transfer the instruction loaded therein for being processed by the task processor only during even time cycles and enabling the second bank of registers to transfer the instruction loaded therein for being processed by the task processor only during odd time cycles.

**Brief description of the drawings**

The above and other objects, features and advantages of the invention will be better understood by reading the following more particular description of the invention in conjunction with the accompanying drawings wherein :

- Fig. 1 is a representation of a tree structure upon which is based the processing performed by the system according to the invention.
- Fig. 2 is a schematic representation of the system according to the invention.
- Fig. 3 is a time diagram representing the interleaved tasks performed by the task processor when one of the tasks is a simple case, a dual load case or a dual process case.
- Fig. 4 is a detailed representation of the system according to the invention represented schematically in Fig. 2.
- Fig. 5 is a representation of the format of the instructions used in the system according to the invention.

**Detailed description of the invention**

A tree structure for a destination address searching according to the present invention is illustrated in Fig. 1, where A is the top of the tree, where intermediate nodes are represented with letters B to K and where leaf nodes are represented by the route to follow for the frame such as RT0 to RT5. When the tree corresponds to the source address searching, the result may be to accept to route this frame or not or to check other parameters. Additional filtering decision may for example be done on upper protocol type.

The left part of the tree starting from A,B ... represents a classical binary tree analyzing one bit position at a time and selecting the left branch when 0 and the right when 1, while the right side A,C ... represents a tree having some binary elements and some more complex elements such as node K which has four branches based on two bits analysis or G which has branches the selection of which is based on a comparison done on several long binary patterns (5 bits in the example).

The embodiments of this invention are explained by means of multiple 16 bits instruction words for each node treatment since it is now an implementation technique very common for the one skilled in the art. Instructions having different length may be built for more complex decision methods or where more than two branches are involved. Nevertheless, the present invention could be realized by any different instruction structure.

Referring to Fig. 2, a system according to the invention comprises a bi-task processing engine 10 connected to an external memory 12 by a standard address bus, data bus and control bus represented schematically by bus 14. Memory 12 contains the instructions to be processed at each node of the tree and is divided into two areas 12-1, 12-2. Area 12-1 contains normal size instructions and is only defined by some defined values on some MSB address bits, while area 12-2 contains dual size instructions. According to the invention, memory 12 contains the instructions of two independent trees.

Processing engine 10 includes a main task processor 16 which can be either a finite state machine or a nano-processor, a task scheduler 18 which generates the thread clocking composed of successive alternate even and odd time cycles and starts processing activity and loading for each task. The task scheduler is connected to the task processor 16 to start a node process and to bank A 20 and bank B 22 by activation lines 24 and 26. Bank 20 and 22 contain respectively the instructions associated with each node in both trees. The activation lines 24 and 26 activate the loading of the instruction from memory 12 to one bank via external bus 14, while they activate the transfer of the instruction from the other bank to task processor 16 via an internal bus 28 for processing said instruction. At a time, a bank has only one of the above access valid such as bus 28 while the other has access to bus 14. This bus connection is inverted at each edge of the thread clock allowing the process of an instruction on one tree while an instruction of the other tree is loaded into its corresponding bank. One tree is the source address tree while the second one is the destination address tree in the address lookup mechanism for which this invention is described.

Temporary registers 30 contain information for each task that should be maintained between two consecutive processing periods of processor 16, especially when the process of an instruction is split into two times cycles. The patterns to be processed by the instructions in processor 16 are provided by a system bus 32 into pattern register A 34 and pattern register B 36.

The timing by the task scheduler 18 of all major steps for both task processing is illustrated in Fig. 3. The thread clock maintained by the task scheduler has a rising edge when process A is activated and a falling edge when the process B starts. The instruction structure knows exactly the duration of its processing which is constant so there is no need to check whether process A is completed or not, similarly for process B.

When the A task is processed, task B has simultaneously its next instruction loaded into bank B. Similarly, when task B is processed, task A has simultaneously its next instruction loaded into bank A resulting in a permanent interleaving process until each task reach a leaf node. If there is a need to process a longer instruction (stored in the second memory area 12-2) that cannot fit within a single cycle, the invention enables to use two cycles for processing such an instruction. Thus, a dual loading of a long instruction is shown in Fig. 3 in a case where no processing is started until the full loading is completed. It would also be possible to start part of the process and to set A or B state bit when the cycle ends and before the end of the cycle, to store intermediate state or results into the temp register 30 dedicated to the corresponding task.

The dual processing of an instruction is also shown in the case of a single size instruction, which stops the loading during one thread until the instruction is fully processed. This mechanism involves the use of the temp register 30 for this task, as the other task will reuse the task processor in the middle of the processing.

At the beginning Bank A 20 is loaded by a pattern that gives as next instruction, the root address for tree A. The stored address for next instruction (root address in the first step but contained in the instruction for further steps) is given by task processor 16 via NEXT ADD bus 40 and driver 42 on bank A.

The address contents are loaded into ADD register 44 via bank data bus 46 as "InsAD" Instruction Address thanks to a BKRD Bank read command activated by NEXTCMD from task processor 16 on line 48 which informs ADD register 44 to load this next address for instruction. NEXTCMD includes the index (IX) (described in Figure 5) used to increment the next address for instruction. ADD register 44 is a register/counter loaded by "InsAD" which is the address of the first word of the instruction to load and whose value is incremented by an integrated counter to load the other words of the instruction. When a rising edge is found on CKA line 50 from Task scheduler 18, the first address corresponding to the initial state of the counter is presented on external add bus 14 with a Ext read command on line 51 in order to access the external memory. A driver 52 is open by Ext read command on line 51 and driver 42 is also open to allow to present LSB bits of the address to load the memory field into the corresponding register of bank A 20 with a bank write command BKWR. As the first address indicates whether the instruction is a single instruction (located in area 12-1) or a dual instruction (located in area 12-2) it is possible to stop the counter at the end of the cycle on the last word that should be downloaded. In case of a dual size instruction, a bit DUAL is set in ADD register 44 which prevents the task scheduler through CKA line 50 lead to reload the address from bank A to ADD register 44 and the counter can continue to load the remaining words of the instruction into bank A during the next cycle. Bank A size is defined to accept a dual size instruction. For each word loaded from memory thanks to read RD and CS chip select of the memory, ADD register 44 selects the appropriate register in bank A by using as register address the LSB bits of the counter value on bank add bus 54 to load the instruction word in the appropriate bank register thanks to a write BKWR to this location. When CKA rising edge occurs again the DUAL bit is tested. If set, the counter continues and loads further words of the instruction until the counter reaches its limit. If not set, ADD register 44 loads the register of bank A selected by NEXT ADD on bus 40 from task processor 16 with a BKWR Read command which allows the InsAd Instruction Address to be put on bank data bus 46 and then stored in ADD register 44.

When instruction is loaded in bank A 20, task scheduler 18 through CKA signal informs task processor 16 to take the first instruction word in bank A. The first register address is put by task processor 16 on add bus 56 and a read done by SEL A command on line 58 is performed on bank A and the first instruction word provided on data bus 60 can be used by task processor 16 which can then run the instruction based on the pattern to compare in A pattern register 34 which may result in loading other instruction words from bank A using the same mechanism. At the end of the processing within the time cycle, either the process is not completed and temporary results are stored in Temp Register 30-1 for task A or the process of the current instruction is finished, Temp register 30-2 being reserved for temporary storage of task B.

In the latter case, when the instruction process is completed, the position (address in bank A) corresponding to the address of the next instruction in bank A is put on next ADD bus 40 using few bits (3 for eight register positions in bank A) in order to give the address for the next loading to ADD register 44.

In the former case, it is necessary to remind that the current process is not finished. Thus, task processor 16 includes two 1-bit A state and B state registers, each one defining the state at the end of the cycle processing which is used when the processing is done on more than one cycle. This bit indicates whether the instruction to process is a continuation of the previous processing activity or a new instruction.

Fig. 5 describes the format of the instructions used in the preferred embodiment. Each instruction contains three main fields which are :

- the instruction itself which is used by the task processor,
- the comparison field which contains the pattern to compare with the A or B pattern at the current position of analysis,
- The next address field which contains the addresses for the possible next instructions.

The instruction itself is generally defined in a single word. It defines the mode of analysis to perform such as one bit, two bits or three bits full comparison resulting in two , four or 8 branches or if a pattern (several bits) comparison using further comparison fields should be performed. It also includes fields defining how many elements are in the comparison field and how many are in the next address field which defines the size of the instruction and let the processor know if an instruction is fully loaded or not.

In full comparison mode, additional fields are defined indicating which is the next address to use for each output case which can be a direct value or an indexed value. There is one such sub-field by possible branch. Fig. 5 shows the case with 4 branches corresponding to a two bits full comparison: branches 00, 01, 10 and 11.

The index (IX) is a 2 bits field which indicates the real address value based on the address given as base address . 00 as index indicates that the address is the address of the indicated next add field while 01 indicates to increment by one the indicated next add field, 10 to increment by 2 and 11 to increment by 3. Thus, a single next add field allows to point onto up to 4 different instruction elements in memory reducing the size of the instruction itself.

The comparison field stores, if any, the pattern(s) to compare to the bits starting at the current position in the A or B pattern field. For each pattern, a sub-field indicates the length of the pattern (Nbbits), a possible mask (Pattern Mask), the next address sub field (direct or indexed) to use or next comparison to perform when match or not match. The index method is the same as what is defined in the instruction field. It should be noticed that, when the link is performed on another comparison field, the index field (IX) is irrelevant.

The next address field contains the list of addresses of the nodes connected to the branches of the current node. Consecutive addresses may be used but cannot always be used as in case of multiple branches, some ones may be followed by a single instruction and some other ones by a dual instruction.

## CLAIMS

1. Interleaved processing system in a network router wherein the frame to be routed includes a header containing a source address pattern and a destination address pattern each of said patterns being processed by a task processor (16) according to a tree structure starting from a root node and being made of a plurality of nodes linked by branches in which each node is associated with an instruction and has at least two child nodes linked to the parent node, the tree processing being performed at each node through the application of the instruction associated with each node and whose parameters depend on said node for determining which branch is to be taken in accordance with said pattern ;

said system being characterized in that it comprises :

- a first bank of registers (20) for loading the instruction to be used by said task processor at each node of the tree corresponding to said source address pattern,
- a second bank of registers (22) for loading the instruction to be used by said task processor at each node of the tree corresponding to said destination address pattern, and
- a task scheduler (18) for generating successive alternate even and odd time cycles and for enabling said first bank of registers to transfer the instruction loaded therein for being processed by said task processor only during even time cycles and enabling said second bank of registers to transfer the instruction loaded therein for being processed by said task processor only during odd time cycles.

2. System according to claim 1, further comprising an address register/counter (44) for storing the address in a memory (12) of the next instruction to be loaded into either first bank of registers (20) or second bank of register (22) before being used by said task processor (16).

3. System according to claim 2, wherein said address of the next instruction to be processed is provided for loading by said task processor (16) into said bank of registers (20 or 22), said address being transferred from said bank of registers to said address register /counter (44) for being used to fetch said next instruction from said memory (12).

4. System according to claim 2 or 3, wherein said address register/counter (44) increments said address of the next instruction when this one is a dual instruction.

5. System according to claim 4, wherein said memory containing said instructions to be used by said task processor (16) comprises a first memory area (12-1) containing normal size instructions and a second memory area (12-2) containing dual size instructions.

6. System according to claim 5, wherein the loading into either first bank or registers (20) or second bank of registers (22) of a dual instruction is interrupted during one time cycle if such a loading requires two time cycles, the time cycle during which said loading is interrupted being used for loading an instruction into the other bank of registers.

7. System according to claim 5, wherein the processing time of said task processor (16) using a dual instruction from either first bank of registers (20) or second bank of registers (22) is interrupted during one time cycle if such a processing requires two time cycles, the time cycle during which said processing is interrupted being used as processing time by said task processor using the instruction provided by the other bank of registers.

8. System according to any one of the preceding claims, further comprising temporary registers (30) for storing information from said task processor (16) between two consecutive processing time cycles when such a processing lasts more than one time cycle.

9. System according to claim 8, further comprising a 1-bit state register for each of said first (20) and second (22) bank of registers, said 1-bit state register being set when said processing lasts more than one time cycle.

# INTERLEAVED PROCESSING SYSTEM BASED UPON THE TREE STRUCTURE IN A NETWORK ROUTER

## Abstract

Interleaved processing system in a network router wherein the frame to be routed includes a header containing a source address pattern and a destination address pattern processed by a task processor (16) according to a tree structure. The system comprises a first bank of registers (20) for loading the instruction to be used by the task processor at each node of the tree corresponding to the source address pattern, a second bank of registers (22) for loading the instruction to be used by the task processor at each node of the tree corresponding to the destination address pattern, and a task scheduler (18) for generating successive alternate even and odd time cycles and for enabling the first bank of registers to transfer the instruction loaded therein for being processed by the task processor only during even time cycles and enabling the second bank of registers to transfer the instruction loaded therein for being processed by the task processor only during odd time cycles.

FIG. 2

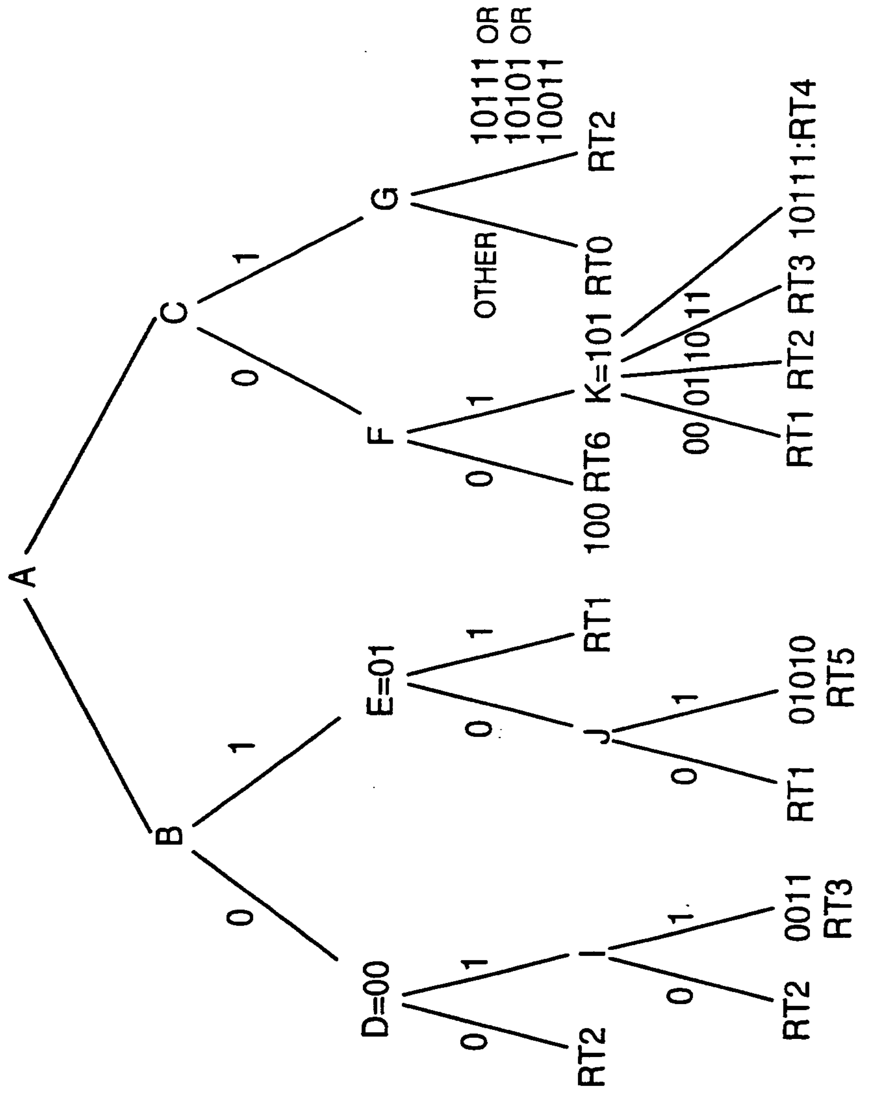

FIG. 1

- 01 : ROUTE RT1
- 00 : ROUTE RT2
- 0011 : ROUTE RT3
- 101 : ROUTE RT4
- 01010 : ROUTE RT5
- 100 : ROUTE RT6
- OTHER : ROUTE RT0 DEFAULT ROUTER

FR 9 99 109
Benayoun et al
2/5

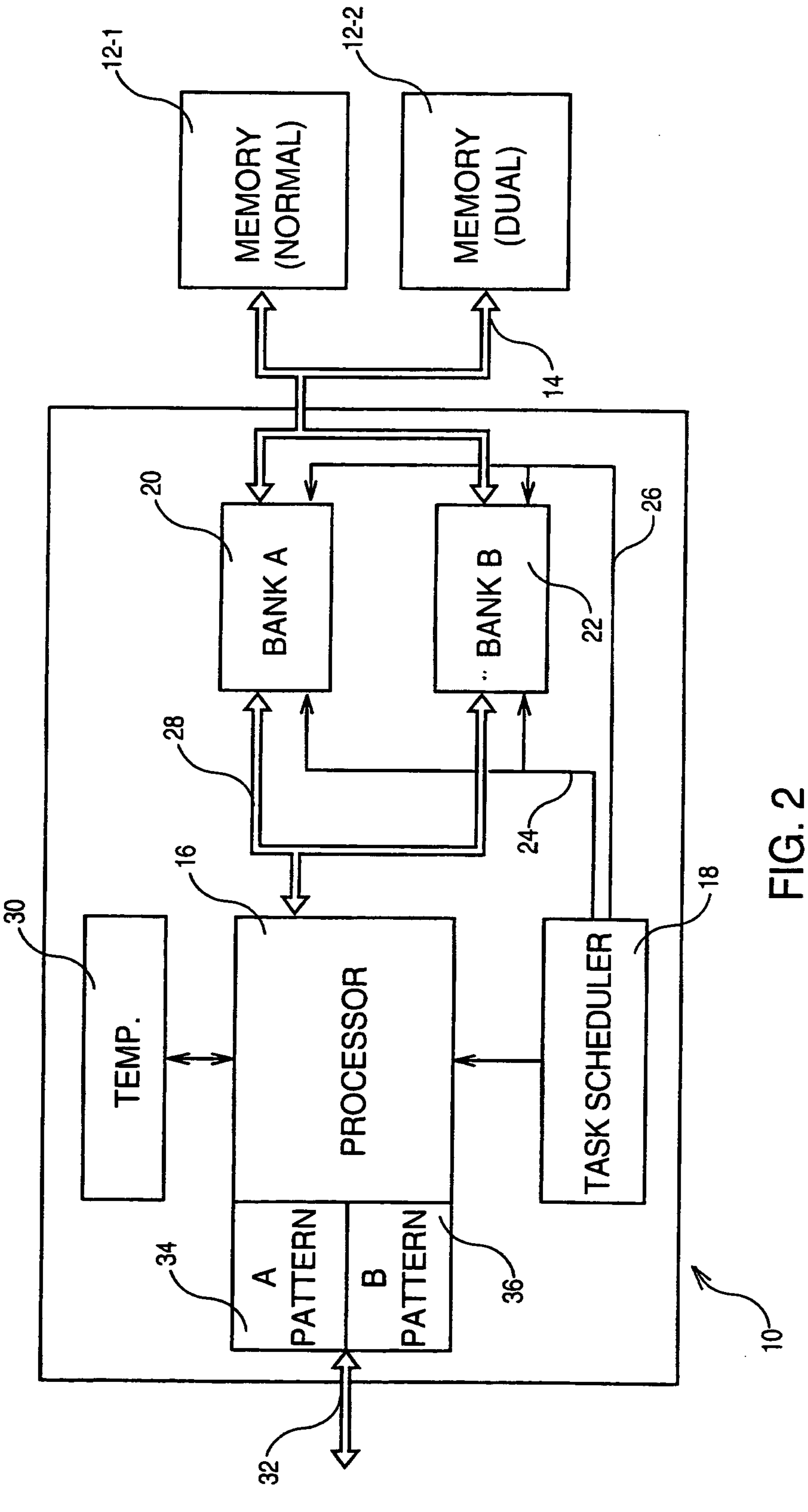

FIG. 2

FR 9 99 109
Benayoun et al
3/5

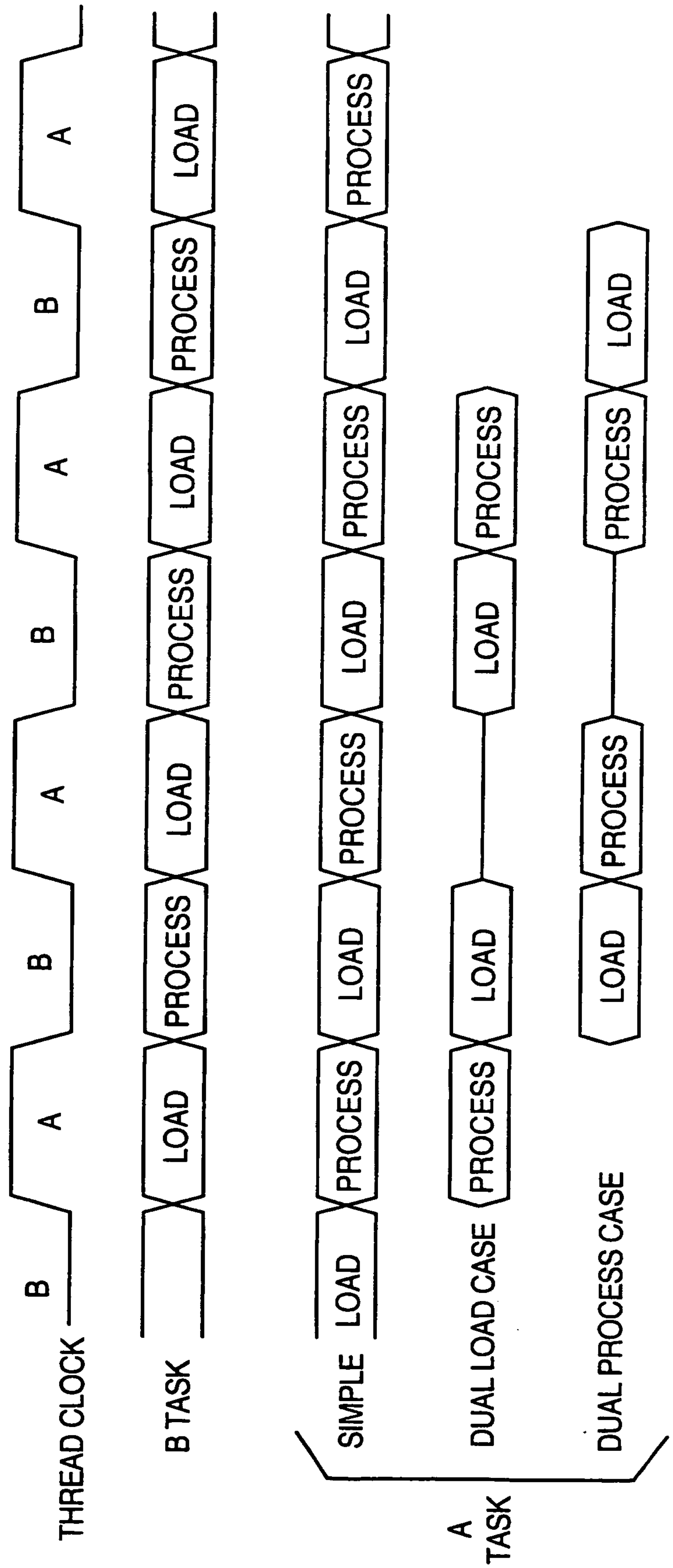

FIG. 3

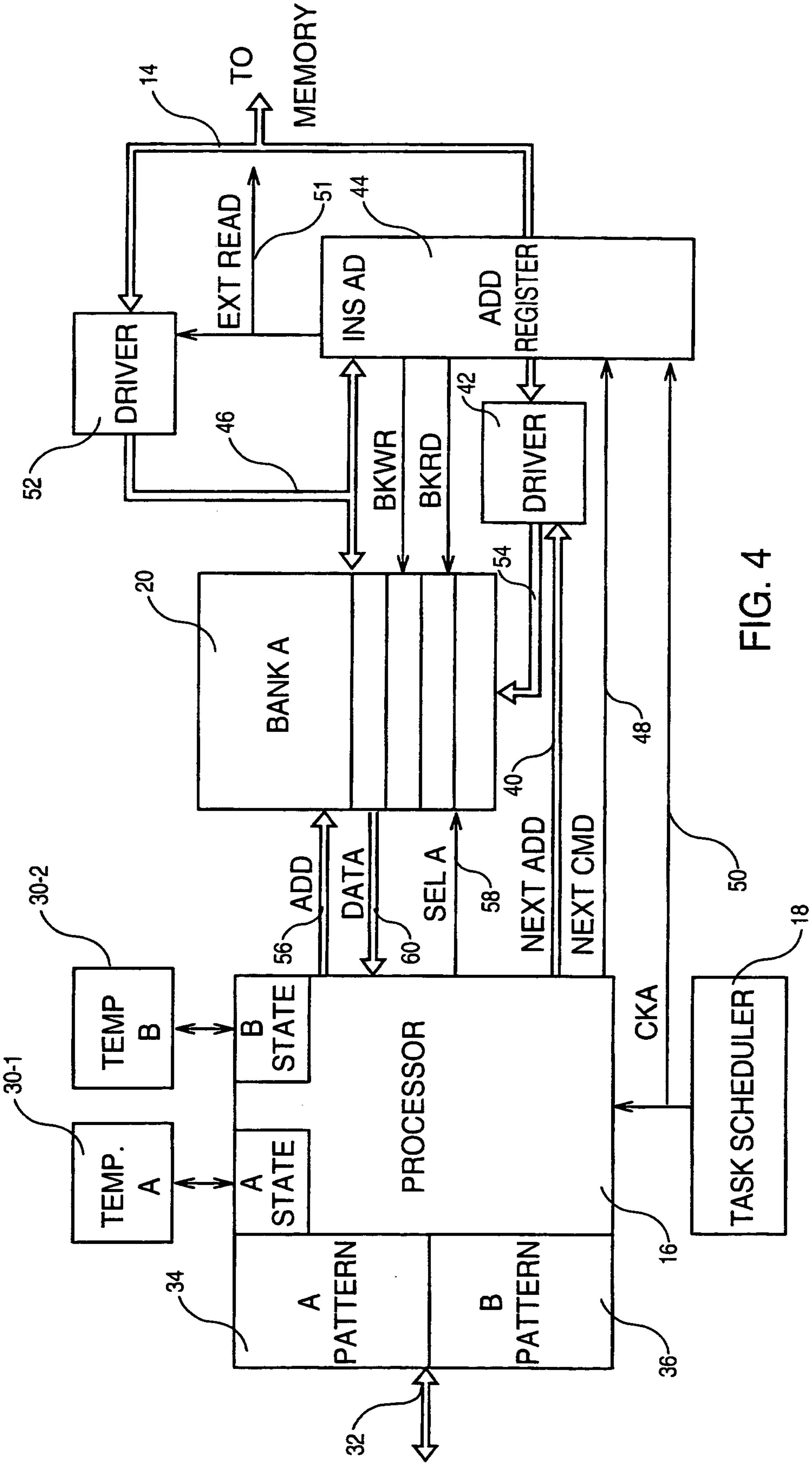
TO MEMORY
14
EXT READ
51
44
INS AD
ADD REGISTER
DRIVER
52
46
BKWR
BKRD
42
DRIVER
20
BANK A
54
40
48
50
30-2
TEMP B
30-1
TEMP. A
B STATE
A STATE
PROCESSOR
ADD
56
DATA
60
SEL A
58
NEXT ADD
NEXT CMD
CKA
18
TASK SCHEDULER
16
34
A PATTERN
B PATTERN
36
32

FIG. 4

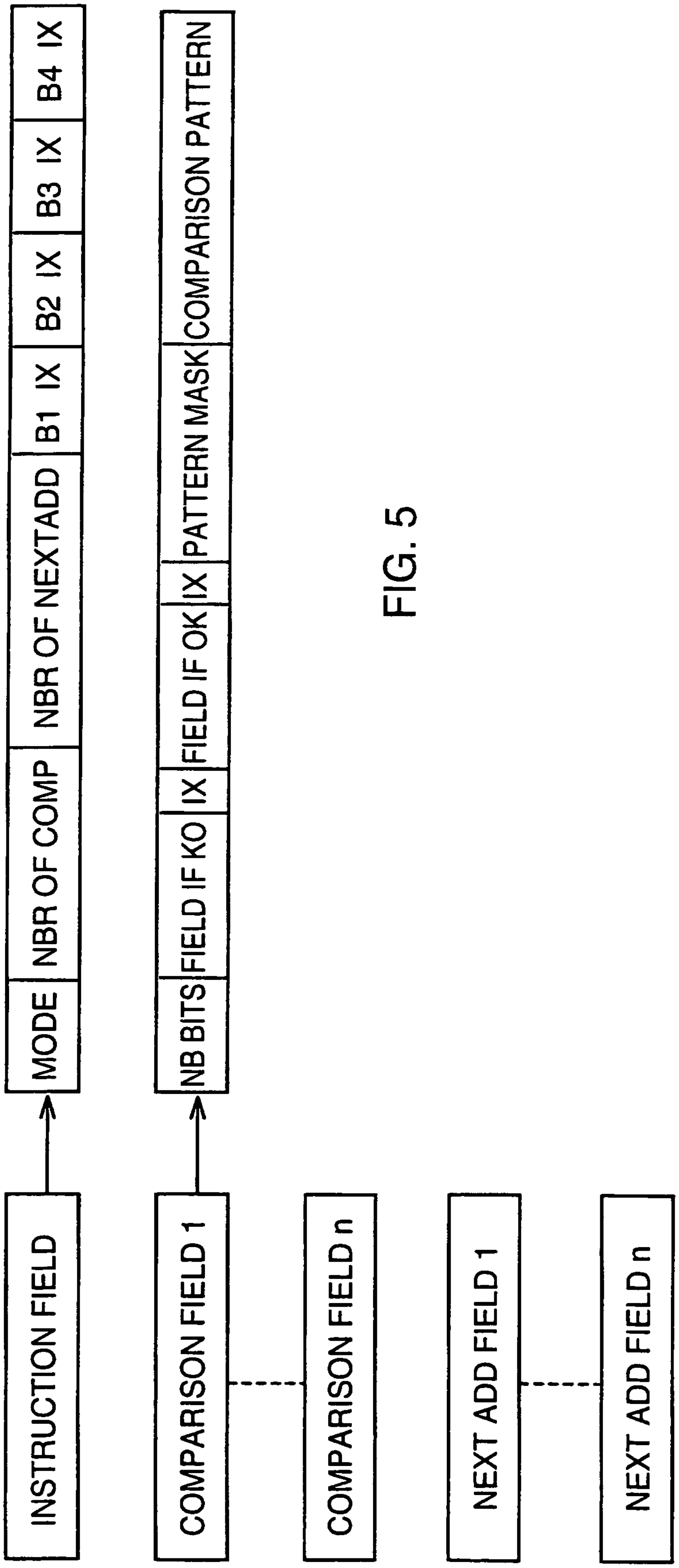
MODE
NBR OF COMP
NBR OF NEXTADD
B1 IX
B2 IX
B3 IX
B4 IX
NB BITS
FIELD IF KO
IX
FIELD IF OK
IX
PATTERN MASK
COMPARISON PATTERN
INSTRUCTION FIELD
COMPARISON FIELD 1
COMPARISON FIELD n
NEXT ADD FIELD 1
NEXT ADD FIELD n

FIG. 5